CLAIMS

1. A computer program embodied on a computer-readable medium providing a development environment for creating a component-based architecture, the computer program comprising:

first software configured to register one or more software components in accordance with an interface definition file associated with each component, each interface definition file being based on a structured markup language identifying one or more methods invocable by the associated component;

second software configured to generate a graphical interface so as to enable a user to build a component-based script in a hierarchal tree format using the components, the hierarchal tree format comprising a plurality of hierarchal levels, wherein at least one instance of a component of a first level is configured to interact with at least one component of a second level as defined by the one or more methods;

third software configured to manage a data structure which organizes the components and the one or more methods of the script; and,

fourth software configured to deploy the script to a component-based architecture.

2. The computer program of claim 1, wherein the first software performs the steps of:

parsing the interface definition file, each interface definition file comprising an interface definition, and

storing the interface definition into a memory module.

3. The computer program of claim 1, wherein the fourth software deploys the script using a deployable interface individually implemented by the one or more components.

4. The computer program of claim 1, further comprising:

fifth software configured for exchanging components over a network.

5. The computer program of claim 4, wherein the network is a global network.

6. The computer program of claim 4, wherein the fifth software is a webserver.

7. The computer program of claim 4, wherein the network is a peer-to-peer network.

8. A computer-implemented method providing a visualization of a component-based architecture development environment comprising the step of:

generating a display of a component-based script in a hierarchal-tree format comprising one or more hierarchal levels, each level comprising at least one component wherein a relationship between the at least one component of a first hierarchal level and the at least one component of a second hierarchal level is visually represented as being defined by a method.

9. The computer-implemented method of claim 8, wherein the display comprises a canvas for displaying the script, and an interacts window for displaying the one or more methods available for defining the relationship between the components of the script.

10. The computer-implemented method of claim 9, wherein the interacts window further displays one or more components that may be passed as a parameter to the one or more method selections.

11. The computer-implemented method of claim 8, wherein portions of the script may be toggled between a hidden state and a revealed state.

12. The computer-implemented method of claim 8, wherein the display of the script may be expanded or collapsed in real time enabling an unlimited number of intermediate positions.

13. The computer-implemented method of claim 8, wherein the one or more components are displayed connected to the method by connect-bars.

14. A software component framework including one or more components, each component comprising:

a component binary comprising an implementation portion of the component framework,

a component wrapper comprising an interface portion of the component framework enabling the component binary to interface with a development environment, and

an interface definition file comprising an interface definition portion of the component framework that enables the component wrapper to register with the development environment.

15. The software component framework of claim 14, wherein the component binary is a compiled object class.

16. The software component framework of claim 14, wherein the interface definition file comprises a description schema in a structured markup language.

17. The software component framework of claim 16, wherein the structured markup language is eXtensible Markup Language (XML).

18. The software component framework of claim 14, wherein the interface definition file comprises a meta section, an interface section, and a component-tree section.

19. The software component framework of claim 14, wherein the interface definition file identifies one or more classes of which the component wrapper is an inheriting subclass.

20. The software component framework of claim 18, wherein the interface section comprises one or more method sections each identifying a method encapsulated in the component wrapper, the one or more methods being an attachment point to the component wrapper.

21. The software component framework of claim 20, wherein each method section further identifies an object type passed to the method at invocation.

22. The software component framework of claim 14, wherein if the one or more components is deployable, the one or more components has an individual build method for a deployment process.

23. The software component framework of claim 14, wherein the one or more components is shared over a network.

24. The software component framework of claim 20, wherein the one or more methods is of a component method-type that sets a new component to a component script.

25. The software component framework of claim 20, wherein the one or more methods is of a property method-type that sets a value to a property defining a component.

26. The software component framework of claim 20, wherein the one or more methods is of a command method-type that denotes action within the component.

27. The software component framework of claim 25, wherein the property method-type sets the value to the property from either a link component whose value is set dynamically or a link-constant component whose value is set statically.

28. A computer system for providing a development environment for creating a component-based architecture, the system comprising:

a processor for registering one or more software components in accordance with an interface definition file associated with each component, each interface definition file being based on a structured markup language identifying one or more methods invocable by the associated component;

a display for displaying a graphical interface so as to enable a user to build a component-based script in a hierarchal tree format using the components, the hierarchal tree format comprising a plurality of hierarchal levels, wherein at least one component of a first level is configured to interact with at least one component of a second level as defined by the one or more methods;

a memory module for holding a data structure which organizes the components and the one or more methods of the script.

29. The computer system of claim 28, wherein the processor deploys the script to a component-based architecture.

30. The computer system of claim 28, wherein the processor performs the steps of:

parsing the interface definition file, each interface definition file comprising an interface definition, and

storing the interface definition into the memory module.

31. The computer system of claim 29, wherein the processor deploys the script using a deployable interface individually implemented by the one or more components.

32. The computer system of claim 28, further comprising a communications interface for exchanging components over a network.

33. The computer system of claim 32, wherein the network is a global network.

34. The computer program of claim 32, wherein the network is a peer-to-peer network.